Introducing Proxy Voting to Helios

Oksana Kulyk*, Karola Marky*, Stephan Neumann*, Melanie Volkamer*†
*Technische Universität Darmstadt, Germany
Email: name.surname@secuso.org
‡Karlstad University, Sweden

***Abstract*—Proxy voting is a form of voting, where the voters can either vote on an issue directly, or delegate their voting right to a *proxy*. This proxy might for instance be a trusted expert on the particular issue. In this work, we extend the widely studied end-to-end verifiable Helios Internet voting system towards the proxy voting approach. Therefore, we introduce a new type of credentials, so-called *delegation credentials*. The main purpose of these credentials is to ensure that the proxy has been authorised by an eligible voter to cast a delegated vote. If voters, after delegating, change their mind and want to vote directly, cancelling a delegation is possible throughout the entire voting phase. We show that the proposed extension preserves the security requirements of the original Helios system for the votes that are cast directly, as well as security requirements tailored toward proxy voting.**

***Keywords*-Electronic Voting, Proxy Voting, Helios**

I. Introduction

Two well-established forms of democracy are *direct democracy*, whereby the decisions on each issue are voted on by the voters directly, and *representative democracy*, where the voters select their representative for a fixed period of time. Both of these forms have their advantages and disadvantages: while direct democracy gives an opportunity to be involved in the decision making process more intensively, frequent voting on every issue might overwhelm the voters, especially if they do not consider themselves to be informed enough. On the other hand, in representative democracy there is a risk that the voters might feel that they are not represented adequately and too disconnected from the government [1]. A hybrid of those two forms, that relies on the *proxy voting* principle, has been proposed. In an election following the proxy voting principle, the voter has the right to either vote herself or delegate her voting right to someone else, such as a trusted expert. Thereby, voters individually have the possibility to decide to which extent they want to directly participate in democratic processes.

Due to the potentially increased voter involvement in democratic processes, Internet based elections could substantially lower voters' burden. For ensuring the security of elections conducted over the Internet, scientific research has resulted in a number of Internet voting systems [2]–[4][1]. Among the most established systems, Helios [2] provides cryptographic end-to-end verifiability, enabling voters and external observers to verify the correctness of the election result, while ensuring vote secrecy at the same time. The system has been used in several real-world elections, e.g. the elections of the International Association for Cryptologic Research [5] or the University president election at UC Louvain [6].

In this work we set as our goal to extend the protocol underlying the Helios voting system with a proxy voting functionality. To achieve this goal, we face several challenges: First, due to the potentially increased frequency of elections, voters should not be required to register for each individual election. The second challenge refers to a proxy who accumulated a lot of *delegation power* – that is, received a significant number of delegations. This constellation is not inherently problematic, but it might lead to a misuse of power, if the number of accumulated delegations for each proxy is known to the public or to a third party. Thus, we require that a proxy should be restricted in her capability to prove how many votes have been delegated to her. Finally, in order to ensure that each voter has full control over her vote, we provide a *cancellation of delegation* functionality. This means that at any point in time before the tallying, the voter should be able to vote directly even if she already delegated her vote.

In our extension, we introduce a new type of credential, so-called *delegation credentials*. These credentials are generated once for each voter, and can be reused in the subsequent elections. The delegation credentials are used by the voters to construct the *delegation tokens* for delegating their vote in each individual election. To do this, tokens are being forwarded to the proxies in an anonymised way and then submitted by the proxies together with the delegated votes during the vote casting phase. The validity of the tokens, and thus the validity of the delegations, is being verified only in the final stage of the election after further anonymisation. In this way, a proxy does not know whether the token she received is valid and is included in the tally. If the voter decides to cancel the delegation and cast her vote directly, all her delegation tokens are marked as invalid. In this way, the delegated votes are discarded from tallying.

The paper is structured as follows: In Section II we list the requirements, including the ones mentioned above, that the proposed protocol shall satisfy. In Sections III and IV we provide the necessary background information by describing the cryptographic primitives used in our work and the protocol underlying the Helios voting system. We present our proxy voting extension in Section V. A security evaluation of our protocol is given in Section VI. Related work is presented in

[1]In this paper we use the term *protocol* to refer to the theoretical proposal, and *system* to refer to the actual implementation

This work has been published in Proceedings of the 11th International Conference on Availability, Reliability and Security (ARES), 2016 , pp. 98-106, IEEE, August 2016. DOI: 10.1109/ARES.2016.38

Section VII. The paper is concluded in Section VIII.

II. Requirements

We describe the requirements, both functional and security-related, that we want our protocol to ensure. We base this list upon the requirements provided in previous work [7].

The following terms will be used in further discussion. We consider *proxies* to be persons, most probably recognized experts, political figures or other opinion leaders, who are registered in the voting system, so that the voters could delegate their votes to them. In this work we do not consider the process of choosing proxies, since it has no influence on the protocol. One possibility would be that everyone can register themselves as a proxy prior to the election. We refer to voters as *delegating* if they delegate their vote to one of these proxies, and *non-delegating*, if they cast their vote by themselves. The voter or proxy is furthermore referred as *honest* if she follows the voting or delegating protocol without violations[2]. A proxy is *semi-honest* if she follows the delegation protocol without violations, yet might try to gain additional information from the data she receives during the protocol execution.

A. Functional Requirements

The goal of our extension is to introduce additional functionality that enables the voters to delegate their vote. While there is no established set of requirements that are considered essential for the proxy voting, we chose to focus on the particular functionality, that we consider to be of use in such an election setting. Namely, we introduce following functions that should be ensured by the system:

a) Delegation: The voter should be able to choose a proxy from a list of available proxies and transfer her voting right in the election to her. This proxy then has the right to vote on behalf of this voter. Note, that we do not place any restrictions on how and whether the proxy should use her delegated voting right: she can vote for any voting option on behalf of the voter, or not cast a vote at all.

b) Cancelling the delegation: After delegating her voting right, the voter should have the option to change her mind and vote herself. Such cancellation must remain possible at any moment of the election prior to the tallying. As we assume that the voter's own vote always has a higher priority than the vote cast for her by a proxy, we do not account for a scenario whereby the voter casts her vote, but changes her mind and wants to delegate later.

c) Back-up delegation: The voter should have the right to delegate her voting right to several proxies, each one of those receives a different delegation priority. In this case, only the vote with the highest priority must be included in the final tally. We consider this functionality to be useful for following reasons. First, in this way the voter can change her mind, if after the delegation she decides to delegate to a different person. Then she delegates her vote again using a higher priority this time. In a second use case, the voter wants to delegate to a particular person, but is not sure whether this person would actually use the delegated voting right and cast the vote. In this case the voter delegates twice: once to the person she would prefer to delegate to, and twice, with a lower priority to some other proxy whose vote would then count if the voter's first choice does not participate.

[2]This excludes the voters who deliberately try to sell their vote.

B. Security Requirements

Security requirements in electronic voting has been an extensive research topic, whereby formal and informal definition requirements have been proposed in the literature [8], [9]. Basing our extension on the Helios protocol, our goal is to preserve the security requirements offered in the original protocol with the addition to the improvements proposed in several extensions [10], [11]. As such, we did not focus on ensuring the coercion resistance requirement, since Helios was designed for use in low-coercion environments. Hence, following requirements should be fulfilled for honest non-delegating voters:

a) Vote secrecy: The voting system should not provide any information to establish a link between the voter and her vote, aside from what is available from the election result.

b) Eligibility: Only the votes cast by eligible voters should be included in the tally.

c) Integrity: Each cast vote of an eligible voter should be correctly included in the tally. In particular, the protocol should provide methods to the voters and to the external observers to verify, that the votes have been *cast as intended*, *stored as cast* and *tallied as stored*.

d) Availability: After the vote casting is finished, the election result can be computed even in case where some of the system components are faulty.

We further outline the security requirements that are specific for the delegation process in particular. We base these requirements upon the security assurances for non-delegating voters.

e) Delegation eligibility: The proxy can only cast their votes on the behalf of eligible voters.

f) Delegation integrity for voters: No proxy can vote on the voter's behalf unless authorised by the voter. Furthermore, if the voter delegates, the proxy cannot alter the priority assigned to her.

g) Delegation integrity for proxies: The valid votes cast by proxies are correctly included in the final tally.

h) Delegation privacy: For honest delegating voters, the identity of the corresponding semi-honest proxy is not leaked. Furthermore, for a given delegating honest voter, a semi-honest proxy should be unable to tell whether this voter delegated to her or to someone else.

i) Vote secrecy for proxies: The voting system should not provide any information to establish a link between the honest proxy and her vote, aside from what is available from the election result.

j) Delegation unprovability: The semi-honest proxy should be unable to provide any additional information to establish her delegating power, aside from what is available from the election result. This means, that the proxy should be unable to prove both before and after the tallying, how many votes have been delegated to her.

Due to the concept of proxy voting, some information about the individual proxy or voter's intention would be leaked from the election result. For example, a group of f colluding voters might attempt to break vote secrecy for proxies and to find out the vote of a given proxy by delegating to her and checking, which voting options have at least f votes for them. Furthermore, even in absence of any attacks, the election result might reveal more information than in the case of non-proxy voting, since the same amount of votes are now being cast by less voters. Such information leakage, however is independent of a particular proxy voting protocol that is being used. Hence, we do not consider this evaluation leakage in evaluating the security of our protocol.

III. Background

In this section we describe the cryptographic primitives used in our protocol.

A. ElGamal

Our extension, as well as the original Helios protocol, builds upon the public-key ElGamal cryptosystem [12] for encrypting the cast votes and auxilirary data. Given $\mathbb{G}_q$ as a cyclic group with prime multiplicative order q where the Decisional Diffie-Hellman is assumed to hold, the public key is defined as (g, h) with $s = \log_h g \in \mathbb{Z}_q$ as a secret key. The encryption of the message $m \in \mathbb{G}_q$ is calculated as $\{m\}_{pk} = (a, b) = (g^r, m \cdot h^r)$ for a randomly chosen $r \in \mathbb{Z}_q$. The decryption of (a, b) is calculated as $m = b \cdot a^{-s}$. The ElGamal encryption has homomorphic properties, being either multiplicatively homomorphic, or additively homomorphic if the value g^m is encrypted instead of m. We denote with $c_1 \cdot c_2 = (a_1, b_1) \cdot (a_2, b_2) = (a_1 \cdot a_2, b_1 \cdot b_2)$ as a pairwise multiplication of ciphertexts c_1, c_2.

B. Zero-Knowledge Proofs

Zero-knowledge proofs are being commonly used for proving the validity of a statement or the knowledge of a witness without revealing any information about the statement or the witness. Examples of such proofs, commonly used in e-voting protocols, are *proof of discrete logarithm knowledge* [13], *proof of discrete logarithm equality* [14] or *knowledge of representation* [15]. We use following notation in this paper. For example, for proving the knowledge of discrete logarithm $x = \log_g h$ for public parameters g, h and secret x, we denote the proof of knowledge π as:

$$\pi = PoK\{x : g^x = h\}$$

A related concept, *signatures of knowledge* [15], describes utilising proofs of knowledge for computing a digital signature on a given message m. Similar to the notation above, we denote such a signature of knowledge on m, for example, using the secret x, public values (g, h) and proof of discrete logarithm knowledge:

$$\pi = PoK\{x : g^x = h\}(m)$$

Cramer et al. [16] describe constructing disjunctive witness-hiding proofs that, among other usage variants, allow proving that a given ciphertext encrypts a message from a given set without revealing which message it is. General methods for constructing proofs of knowledge for binary formulas of statements about discrete logarithms were suggested by Camenish et al. [17], which we also use for constructing proofs in our protocol.

For making the proofs/signatures of knowledge non-interactive, a strong version of the Fiat-Shamir heuristic [18] is applied.

C. Threshold Verifiable Decryption

In order to avoid a single point of failure by trusting a single entity concerning the secret key for the cryptosystem, protocols that enable sharing the key among several entities have been developed. In particular, the protocol by Pedersen [19] enables distributive key generation and sharing between multiple entities in a verifiable manner.

A method for using a key shared in this manner for distributively decrypting the ciphertext is proposed, whereby the decryption succeeds if at least a threshold t of total N entities ($t > N/2$) participate in the protocol. The zero-knowledge proofs used for the decryption ensure that the ciphertexts have been decrypted correctly without manipulation.

D. Mix Net

The mix net shuffle is used for anonymising the ciphertexts $c_1, ..., c_N$ by permutating and reencrypting the input list. In that way, a new list of ciphertexts $c'_1, ..., c'_N$ is produced, with $c'_i = c_{\pi(i)} \cdot (g^{r_i}, h^{r_i})$, π denoting a permutation on $\{1, ..., N\}$, (g, h) a public encryption key for $c_1, ..., c_N$, r_i a random value. In this way, as long as π and $r_1, ..., r_N$ remain secret, the link between the ciphertexts in the input and output lists cannot be established.

In order to ensure, that no manipulation occured during the shuffle, and the ciphertexts in the input and output list encrypt the same plaintexts, a number of *proof of shuffle* techniques have been proposed in the literature. In particular, the most efficient proofs with a high soundness have been proposed in [20], [21]. Note, that these proofs can also be adjusted for shuffling tuples of ciphertexts $\bar{c_1}, ..., \bar{c_N}$ with $\bar{c_i} = (c_{i,1}, ..., c_{i,k})$, so that the ordering within the tuples is preserved in the output list.

IV. Helios System

The Helios system and its underlying protocol have been extensively studied in literature [18], [22], [23]. To base our protocol we chose Helios-C [10], which introduces voter credentials using a public-key infrastructure, with further modification of the original system that uses a threshold verifiable

decryption [11]. For the sake of simplicity, we describe the single choice ("yes/no") election, where the voters cast either 1 or 0 (represented as g^0 or g^1) as their vote, although a generalisation to more complex ballots is possible. The following entities are involved in the protocol:

- *Registration authority*, responsible for generating and distributing the credentials to eligible voters,
- *Bulletin board* acting as a public append-only broadcast channel that is used for publishing all necessary election information and cast votes,
- *Tabulation tellers*, responsible for generating the election keys, anonymising the cast votes and decrypting the result.

The election process can be briefly outlined as follows.

a) Setup: The registration authority distributes the voting credentials to eligible voters and publishes the corresponding signing public keys on the bulletin board. In further descriptions we imply that everything published on the bulletin board by the voters is signed using the corresponding signing key. Using the method for threshold key generation, the tabulation tellers generate a pair of ElGamal keys $pk = (g, h = g^s) \in \mathbb{G}_q^2, sk = s \in \mathbb{Z}_q$ for the election, that are later used for encrypting and decrypting the cast votes. The public election key $pk = (g, h)$ is published together with the list of valid voting options $\{o_1, ...o_L\} \subset \mathbb{G}_q$.

b) Voting: In order to cast a vote for a voting option $o \in \{o_0 = g^0, o_1 = g^1\}$, the voter V_i authenticates herself to the bulletin board. Then she prepares and signs her ballot $(v = \{o\}_{pk}, \pi_v)$ with:

- $v = (a, b) = (g^r, oh^r)$ as the encryption of a voting option o using a public key pk,
- $\pi_v = PoK\{r \in \mathbb{Z}_q : a = g^r \wedge (b = o_0 h^r \vee b = o_1 h^r)\}$ the zero-knowledge proof of well-formedness, used to prove the plaintext knowledge of o[3] and that $o \in \{o_0, o_1\}$ is a valid voting option.

After preparing a ballot, the voter has an option either to cast it as her vote by submitting it to the bulletin board, or to audit the ballot using the Benaloh challenge [24]. The purpose of the audit is to ensure, that the ballot was prepared correctly by the voting device, ensuring cast-as-intended verifiability. The voter can audit as many ballots as she wants, until she is ready to cast her vote. After casting the vote, the voter checks whether it has been correctly posted on the bulletin board, ensuring stored-as-cast verifiability.

c) Tallying: After the vote casting phase has finished, the bulletin board removes all duplicate ballots and ballots with invalid signatures or zero-knowledge proofs. In case vote updating is allowed, among the votes cast with the same voter credential, only the last ballot is kept. Prior to the decryption, the votes have to be anonymised, in order to remove the link between the chosen voting options and the voters' identities. There are two ways to perform this anonymisation, both are used in various versions of the Helios system. The first way is to use a verifiable mix net shuffle with the each tabulation teller acting as a mix node. The second way is to make use of the homomorphic properties of the ElGamal cryptosystem and aggregate the votes to a homomorphic sum, which represents the final result. Note, that the latter option requires additional considerations regarding the encoding of voting options depending on the ballot type.

After the votes have been anonymised, the result of the anonymisation is verifiably decrypted by the tabulation tellers and published. The zero-knowledge proofs of tally correctness, calculated during the mix net and the encryption, as well as signatures and zero-knowledge proofs submitted with the ballots during vote casting, are published to enable the tallied-as-stored verifiability of the election result.

[3]This is required to ensure the non-malleability of ballots and prevent ballot copying attacks [18].

V. Proxy Voting Extension

In this section we show how to extend the Helios system towards proxy voting.

A. Setup

The initial setup is performed analogously to the original protocol. We assume following additional infrastructure in our extension. For each voter V_i we assume the existence of T public *delegation credentials*, represented as an ordered tuple $h_{i,1}, ..., h_{i,T}$. These credentials are posted by the registration authority on the bulletin board, whereby the voter knows the secret keys $x_{i,j} = \log_g h_{i,j}$. The credential can either be generated and sent to the voters by trusted parties, or uploaded by the voters themselves, whereby they choose the values of $x_{i,j} \in_R \mathbb{Z}_q$ at random and submit $h^{x_{i,j}}$ to the bulletin board prior to the election. Furthermore, the list of available proxies $D_1, ..., D_n$ and their public signing keys is made available as well, and communication channels that enable voters to securely communicate with proxies are established. As in the description of the original protocol, we imply that everything that the voters or proxies publish on the bulletin board is signed by their corresponding signing key.

B. Vote Casting

a) Voting: Casting a vote is the same as in the original Helios. The voter authenticates herself to the bulletin board, and submits the ballot of the form (v, π_v), which is then published on the bulletin board. For ensuring cast-as-intended verifiability, she can also choose to audit her ballot instead of casting it.

b) Delegating: For delegating with priority $j = 1, ..., T$, the voter V_i computes an encryption of her credential $c = (a_d, b_d) = (g^{r_d}, h_{i,j} h^{r_d})$, a commitment $\sigma = g^m$ of a randomly chosen value $m \in \mathbb{Z}_q$ and a non-interactive signature of secret key knowledge on σ, $\pi_d = PoK\{(r_d, x_{i,j}) : a_d = g^{r_d} \wedge b_d = g^{x_{i,j}} h^{r_d}\}(\sigma)$ (see Algorithm 1). The *delegation token*, which are the values (σ, m, c, π_d) are then sent to a proxy of the voter's choice over a private anonymous channel[4].

[4]The public-key infrastructure encompassing the proxies can be used to ensure the privacy of the communication, and an anonymous communication network such as TOR [25] can be used for anonymity.

Algorithm 1 Signature of knowledge of valid delegation token for priority j

Private input: $m, r \leftarrow_R \mathbb{Z}_q$, $h_j \in \mathbb{G}_q$, $x_j = \log_g h_j \in \mathbb{Z}_q$
Public Input: $(g,h), c = (a_d, b_d) = (g^{r_d}, h_j h^{r_d}) \in \mathbb{G}_q^2$, $\sigma = g^m \in \mathbb{G}_q$
Proof:
$w_1, w_2 \leftarrow_R \mathbb{Z}_q$, $t_1 \leftarrow g^{w_1}$, $t_2 \leftarrow g^{w_2} h^{w_1}$
$e \leftarrow H(\sigma||g||h||a_d||b_d||t_1||t_2)$, $s_1 \leftarrow w_1 - e r_d$, $s_2 \leftarrow w_2 - e x_j$
$\pi_d \leftarrow (t_1, t_2, s_1, s_2)$
Verification:
$e \leftarrow H(\sigma||g||h||a_d||b_d||t_1||t_2)$
if $a_d^e g^{s_1} = t_1 \wedge b_d^e g^{s_2} h^{s_1} = t_2$ **then**
  $\mathsf{Verify}(\pi_d) = 1$
else
  $\mathsf{Verify}(\pi_d) = 0$
end if

c) Casting a delegated vote: The proxy encrypts her chosen voting option as a ciphertext $v = (a_v, b_v) = (g^{r_v}, o h^{r_v})$. She further calculates the zero-knowledge proof $\pi_v = PoK\{r_v, m \in \mathbb{Z}_q : \sigma = g^m \wedge a_v = g^{r_v} \wedge (b_v = o_0 h^{r_v} \vee b_v = o_1 h^{r_v})\}$, which serves both as a proof of well-formedness for v and as a proof of knowledge of a decommitment value m. The proof is described in Algorithm 2. She can then choose to either cast or audit the ballot. The auditing is the same as in the original protocol; if the proxy decides to cast, she submits $(\sigma, v, \pi_v, c, \pi_d)$ as her signed ballot.

Algorithm 2 Proof of valid delegated vote for an option o_i with delegation token (σ, m, c, π_d)

Private input: $m \in \mathbb{Z}_q$, $r_v \leftarrow_R \mathbb{Z}_q$, $i \in \{0,1\}$, $j = \bar{i}$
Public Input: $(g,h), c, v = (a_v, b_v) \in \mathbb{G}_q^2$, $\pi_d \in \mathbb{G}_q^2 \times \mathbb{Z}_q^2$, $\sigma = g^m$
Proof:
$e_j \leftarrow_R \mathbb{Z}_q$
$w_0, w_1, \hat{w}_0, \hat{w}_1 \leftarrow_R \mathbb{Z}_q$
$t_{1,i} \leftarrow g^{w_i}$, $t_{2,i} \leftarrow h^{w_i}$
$t_{1,j} \leftarrow g^{w_j} a_v^{e_j}$, $t_{2,j} \leftarrow h^{w_j} (b_v o_j^{-1})^{e_j}$
$\hat{t}_i \leftarrow g^{\hat{w}_i}$, $\hat{t}_j \leftarrow g^{\hat{w}_j}$
$e \leftarrow H(\sigma||g||h||a_v||b_v||t_{1,0}||t_{2,0}||\hat{t}_0||t_{1,1}||t_{2,1}||\hat{t}_1)$
$e_i \leftarrow e - e_j$, $s_i \leftarrow w_i - e_i r_v$, $\hat{s}_i \leftarrow \hat{w}_i - e_i m$
$s_j \leftarrow w_j$, $\hat{s}_j \leftarrow \hat{w}_j$
$\pi_v \leftarrow (t_{1,0}, t_{2,0}, \hat{t}_0, t_{1,1}, t_{2,1}, \hat{t}_1, s_0, \hat{s}_0, e_0, s_1, \hat{s}_1, e_1)$
Verification:
if $\mathsf{Verify}(\pi_d) \neq 1$ **then**
  $\mathsf{Verify}(\pi_v) = 0$
else
  $e \leftarrow H(\sigma||g||h||a_v||b_v||t_{1,0}||t_{2,0}||\hat{t}_0||t_{1,1}||t_{2,1}||\hat{t}_1)$
  if $e_0 + e_1 = e \wedge \sigma^{e_0} g^{\hat{s}_0} = \hat{t}_0 \wedge \sigma^{e_1} g^{\hat{s}_1} = \hat{t}_1 \wedge a_v^{e_j} g^{s_j} = t_{1,j} \wedge (b_v o_0^{-1})^{e_0} g^{s_0} = t_{2,0} \wedge (b_v o_1^{-1})^{e_1} g^{s_1} = t_{2,1}$ **then**
    $\mathsf{Verify}(\pi_v) = 1$
  else
    $\mathsf{Verify}(\pi_v) = 0$
  end if
end if

d) Cancelling a delegation: If the voter decides to cancel the delegation and vote herself, she just casts her own vote as in the original protocol.

C. Tallying

After the vote casting stage has finished, all duplicate ballots and ballots with invalid proofs or signatures are removed. The updated votes by the non-delegating voters are processed according to the vote updating policy analogously to in the original system, as well as the votes cast by proxies which contain duplicate delegation tokens. These votes are further used to initialize different sets which are required for the tallying process.

Let $V_{own} = \{(v_i, id_i)\}$ be the set of valid votes which were cast by voters directly with corresponding voter IDs. $V_d = \{(v_i, c_i)\}$ denotes the set of valid votes which were cast by proxies, $H = \{h_{1,1}, ..., h_{1,T}, ...h_{N,1}, ..., h_{N,T}\}$ a set of all valid delegation credentials. Before the beginning of the tallying, two sets are initialised: a set $V = \{v : \exists (v, id) \in V_{own}\}$ representing the votes that will be included in tallying, and $H_{own} := \{h_{i,j} : \exists (v, id_i) \in V_{own}\}$ as the list of all delegation credentials of voters who cast their vote themselves.

Following procedure is being executed: The votes cast by proxies $(v_k, c_k) \in V_d$ are being processed through the verifiable mix net shuffle, resulting in an anonymised list of tuples (v'_k, c'_k). After the anonymization, the values of c'_k are decrypted to reveal the delegation credentials h'_k used in constructing the delegating tokens. The votes with $h'_k \notin H$ are discarded as cast with non-valid delegation tokens. The rest of v'_k is assigned to the corresponding delegation credential $h_{i,j}$ with i denoting the voter, and j the registration priority.

After this procedure, V_d should consist of delegated votes with valid delegation credentials $(v'_k, h_{i,j})$. The delegated votes, that were overwritten either by the voter herself, or by a delegated vote with the higher priority, should be discarded. For this, each vote v'_k from the tuple $(v'_k, h_{i,j})$ is added to V if and only if following conditions hold:

1) $h_{i,j} \notin H_{own}$ meaning that the delegated vote is not revoked by the voter via voting directly;
2) $\forall (v'', h_{i,l}) \in V_d : l < j$, meaning that the delegated vote is not overwritten with a delegation of higher priority. Note, that this implies that the votes cast for the same voter with the same delegation priority but different delegation token are not included into the final tally.

The votes in V are being tallied as in the original system: anonymized using either mix net or homomorphic sum, and decrypted to reveal the final election result.

VI. Security of the Extension

In order to evaluate the security of our protocol, we first provide the list of necessary assumptions. The security requirements given in Section II-B are then evaluated given these assumptions.

A. Assumptions

We list the security assumptions required for the security of our protocol. Note, that here we do not consider the assumptions used for ensuring the security of the original protocol, that are not relevant to the delegation process.

(A1) The channels between the honest voters and the proxies are private.
(A2) The channels between the honest voters and the proxies are anonymous.
(A3) At most t of tabulation tellers are corrupted by an adversary.
(A4) The voting devices of both proxies and voters do not leak information to an adversary.
(A5) No coercion or vote selling takes place.
(A6) The adversary is computationally restricted, the DDH assumption holds and the random oracle is instantiated by a hash function.
(A7) The registration authority is trustworthy.

B. Evaluation

We discuss the requirements defined in Section II-B, as fulfilled in this extension.

a) Vote secrecy: As vote casting remains the same as in the original protocol, no information about the individual votes is leaked at this stage, under the condition that the original protocol is secure. The resulting votes are anonymised together with the votes from proxies, which ensures vote secrecy as long as this anonymisation is performed correctly. Since the procedure of the anonymisation does not differ from the original protocol, vote secrecy for non-delegating voters is preserved under the same assumption.

b) Eligibility: Since the voter registration process is not changed, the eligibility is preserved under the same assumptions as in the original protocol.

c) Integrity: As the process of casting a vote is not changed, the voter has cast-as-intended and stored-as-cast verifiability. The tallied-as-stored verifiability depends on the integrity of the anonymisation process, hence it is guaranteed as long as the original protocol is secure.

d) Availability: Due to the threshold decryption approach, as long as at least t out of N tabulation tellers participate in the tallying process, which is given due to the assumption (A3) for $t > N/2$, the final result can be computed.

e) Delegation eligibility: Given the assumption (A7), all the published delegation credentials belong to eligible voters. As long as the decryption of the credentials in delegation tokens is performed correctly, which is ensured by the corresponding zero-knowledge proofs together with the assumption (A6), everyone can verify that only the delegated votes with valid delegation credentials are included in the tally.

f) Delegation integrity for voters: Let us consider the case where the proxy is willing to cast a vote using the credential $h_{i,j}$ on behalf of some voter V_i. For this she needs to calculate the signature of knowledge $\pi_d = SoK\{(r, x_{i,j}) : a = g^r \wedge b = g^{x_{i,k}} h^r\}(\sigma)$, which according to the assumption (A6) she cannot do without the knowledge of $x_{i,j}$. The same argument holds for a proxy who wants to cast her vote with a different priority than the one delegated to her. That is, upon getting the delegation token for the credential $h_{i,j}$, she wants to cast a vote using the credential $h_{i,k}$ for $k \neq j$. Again, given the assumption (A6), she cannot do this without the knowledge of $x_{i,k}$.

An adversary might also attempt to reuse the delegation token once posted on the bulletin board by the proxy and thus update the legitimate proxy's vote. Due to the assumption (A6) and fact that σ is integrated into π_d, she would need to know the value of m in order to calculate the zero-knowledge proof π_v. This, however, is prevented given that m is sent to the legitimate proxy over a private channel according to assumption (A1).

g) Delegation integrity for proxies: Similarly to the votes of non-delegating voters, the votes of proxies would be correctly included in the election result, as long as the proxies perform the audits and check that their votes are published on the bulletin board at the end of the election, and the mix net shuffling and decryption is performed correctly. This is ensured by corresponding proofs together with the assumption (A6).

h) Delegation privacy: Obviously, some information leakage is unavoidable if a given proxy does not get any votes delegated to her, or if a given voter does not appear in the list of delegating voters, either by casting the vote herself or abstaining from the election. This should not be considered to be a violation of privacy. Hence, we consider the following expression for delegation secrecy: Given two delegating voters V_1, V_2, and two semi-honest proxies D_1, D_2 each receiving a delegation token from one of them, D_1 and D_2 should be unable to distinguish between $((V_1, D_1); (V_2, D_2))$ and $((V_1, D_2); (V_2, D_1))$, with (V_i, D_j) denoting the voter V_i delegating to the proxy D_j.

Given the assumption (A4), only information that is either public or sent privately to the proxies could be potentially used for breaking vote secrecy. Furthermore, given the assumption (A5), the voters do not provide any additional information that is not part of the protocol, that might assist in revealing their identity. The assumptions (A1) and (A2) prevent the proxies from using the communication channels for finding out either the identity of the voters who delegated to each of them, or whether V_1 or V_2 communicated with the other proxy.

Consider the proxies D_1, D_2 casting a vote $(\sigma_i, v_i, c_i, \pi_{v,i}, \pi_{d,i})$ with $(\sigma_i, m_i, c_i, \pi_{d,i})$ as the delegation token, $i = 1, 2$. As they are semi-honest, they only have the data available during the protocol run. They have access to encrypted credentials c_1, c_2, the published delegating credentials from both voters $h_{1,1}, ...h_{1,T}, h_{2,1}, ..., h_{2,T}$, and the reencrypted ciphertexts resulting from the mix net shuffle of delegating credentials $(v'_1, c'_1), (v'_2, c'_2)$.

In order to distinguish between a delegation from V_1 or V_2, the proxies D_1, D_2 need to be able to tell,

- whether c_1 and c_2 encrypt $h_{1,k}$ respectively $h_{2,l}$ or vice versa for some $1 \leq l, k \leq T$, OR
- whether (v_1, c_1) and (v'_1, c'_1) (respectively, (v_2, c_2) and (v'_2, c'_2) encrypt the same plaintexts, OR
- whether (v_1, c_1) and (v'_2, c'_2) (respectively, (v_2, c_2) and (v'_1, c'_1) encrypt the same plaintexts.

Unless they have access to the decryption key or the randomness used for reencrypting $(v'_i, c'_i), (v'_j, c'_j)$, the IND-CPA security of the encryption protocol and the zero-knowledge property of the proof π_d restricts them from making the distinction. Thus, given the assumptions (A1-A5), secrecy against semi-honest proxies is preserved.

Same argument can be made for secrecy protection against external adversaries.

i) Vote secrecy for proxies: The votes cast by proxies are encrypted until the final anonymisation, either by a mix net or homomorphic tallying, and subsequent decryption. Thus, similar to vote secrecy in the original protocol, unless the adversary is capable of manipulating the voting device, corrupting at least t out of N tabulation tellers, or breaking the encryption (assumptions (A3), (A4), (A6)), the adversary gets no information about the individual proxy's selection.

j) Delegation unprovability: The proxy knows how many delegation tokens she has gotten in the election – however, without being able to distinguish, whether a given delegation token contains an encryption of a valid delegation credential (assumption A6), she does not know exactly how many of these tokens are actually valid delegations. Furthermore, a third party, unless it has control over the communication channels between the proxy and the voters (which contradicts the assumptions (A1) and (A2)), does not know which ones of the delegation tokens were sent to the proxy, and which she created herself in order to cheat about her delegating power.

After the tallying, however, the total number of invalid delegation tokens $\tilde{N}$ cast within the election is revealed. In this way, if the proxy received or presented to a third party N delegation tokens, it can be concluded that at least $N - \tilde{N}$ of them are valid. Furthermore, if the rest of the proxies are dishonest and do not use their valid delegation tokens to cast a vote, the number of valid delegation tokens corresponds to a delegating power of a proxy. Thus, the requirement of delegation unprovability is only probabilistically ensured, which can be corrected if a sufficient number of "chaff" fake delegations are added to the tally similar to the suggestion in the Civitas system [3].

VII. Related Work

In this section we describe related work done both regarding extension to the Helios system, as well as in cryptographic protocols and implementations for proxy voting elections.

A. Helios Extensions

Several extensions and security improvements have been proposed for the Helios system. The extension by Cortier et al. introduces verifiability against malicious bulletin board by distributing signature credentials to the voters during the registration and requiring the voters to sign their ballots upon vote casting [10]. A further modification [11] introduces distributed tallying via Pedersen secret sharing. The BeleniosRF protocol [26], as well as the protocol by Kulyk et al. [27], [28] introduce receipt freeness into Helios, while the latter protocol also ensures eligibility verifiability with hiding the information on whether a particular eligible voter participated in the election or abstained. The extension ensuring long term privacy in Helios was proposed by Demirel et al. [29].

The Zeus voting system [30], used in University of Athens election, modifies Helios by introducing an additional way to ensure cast-as-intended verifiability. The voters have an option to cast a vote using audit codes distributed to them at the registration, so that the votes cast with those codes are not included in the tallying, but decrypted instead, so that the voters could verify their correctness. Further methods to improve cast-as-intended verifiability in Helios-like protocols have been proposed in the Selene protocol [31], which introduced tracking number appended to the cast votes, and Guasch et al. [32], [33] employing designated-verifier proofs. For boardroom voting, an Android application extending and implementing the protocol behind Helios was developed [34]. From the usability perspective, a number of suggestions for the improvements of the current Helios implementation that simplify the audit process for the voters, have been proposed [35], [36].

B. Proxy Voting

There are a few proxy voting implementations which are published by different organizations. Two widely known systems are LiquidFeedback[5] and Adhocracy[6]. However both approaches completely relinquish vote secrecy, since all actions of users are visible to other users of the system at any time.

Furthermore, a number of protocols were made to conduct cryptographic proxy voting [37], [38]. The first protocol [37] introduces delegation to different proxies with different priorities by using hash chain elements as credentials. Voting is based on submitting an anonymous credential over an anonymous channel. This approach, however, does not offer verifiability for external observers and does not allow for vote updating or voter credentials reuse. In the second protocol [38], the voters either encrypts and casts the vote published by their chosen proxy at the beginning of the election, or cast the encrypted name of their chosen proxy as their vote. Hence, vote secrecy for the votes cast by proxies is not ensured. Both of these protocols ensure vote secrecy for voters and verifiability by requiring stronger assumptions than our protocol, such as reliance on a single trusted entity.

VIII. Conclusion

As new forms of voting are being developed, also secure solutions for these new forms need to be developed. In our work, we present an extension of the well-known Helios voting system that introduces functionality for proxy voting.

This new functionality enables voters to delegate their vote to a trusted proxy while they can change their mind and vote themselves at any moment of the election before the tally. Our protocol secures the delegation process by ensuring that the voter's choice of a proxy and the proxy's vote are private, and that no proxy can cast a vote unless being authorised

[5]http://liquidfeedback.org/

[6]https://adhocracy.de/

by an eligible voter. It furthermore prevents the proxy from proving how many valid delegated votes she has received, and ensures that all the delegated votes from eligible voters, that have not been overwritten by a direct vote or by a delegated vote with a higher priority, are correctly included in the tally. The extension also preserves the security requirements of the original Helios protocol.

As future work, we intend to establish formal definitions of new requirements for proxy voting and provide formal security proofs for our protocol. We further plan to consider different election settings for proxy voting, where a different set of requirements need to be ensured. For example, in the settings where the voters can delegate their votes to a proxy not just for a single election, but also for a number of elections within a given timespan, a possibility for the voters to verify, how their chosen proxy has voted, might be of use.

Another possible direction of future work would be to address the issue of board flooding in our protocol: currently, if the adversary succeeds in casting a large number of delegations with fake credentials, she can exploit this opportunity to slow down the tallying. This issue is also present in a family of other protocols that rely on anonymous channels for casting the ballot [3], [39], and extensions designed to solve this problem has been proposed [40], [41]. However, these extensions alter the adversarial model by requiring additional trust assumptions, and neither of them is designed for proxy voting. Hence, an appropriate solution for our protocol needs to be designed.

We furthermore plan to investigate the usability of our protocol. Since the voting process for non-delegating voters does not differ from the original Helios, we would focus our research on the usability of the delegation process.

ACKNOWLEDGMENT

The research that led to these results has been funded from a project in the framework of Hessen ModellProjekte (HA project no. 435/14-25), financed with funds of LOEWE Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

REFERENCES

[1] S. Coleman, *Direct representation: Towards a conversational democracy*. ippr, 2005.

[2] B. Adida, "Helios: Web-based open-audit voting." in *USENIX Security Symposium*, vol. 17, 2008, pp. 335–348.

[3] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *Proceedings of the 2008 IEEE Symposium on Security and Privacy*. IEEE, 2008, pp. 354–368.

[4] F. Zagórski, R. T. Carback, D. Chaum, J. Clark, A. Essex, and P. L. Vora, "Remotegrity: Design and use of an end-to-end verifiable remote voting system," in *Applied Cryptography and Network Security*. Springer, 2013, pp. 441–457.

[5] IACR, "IACR Election 2013," http://www.iacr.org/elections/2013/, 2013, [Online; accessed 18-February-2016].

[6] B. Adida, O. De Marneffe, O. Pereira, J.-J. Quisquater *et al.*, "Electing a university president using open-audit voting: Analysis of real-world use of helios," *EVT/WOTE*, vol. 9, pp. 10–10, 2009.

[7] O. Kulyk, S. Neumann, K. Marky, J. Budurushi, and M. Volkamer, "Coercion-resistant proxy voting," in *ICT Systems Security and Privacy Protection*. Springer, 2016, pp. 3–16.

[8] S. Neumann and M. Volkamer, "A holistic framework for the evaluation of internet voting systems," *Design, Development, and Use of Secure Electronic Voting Systems*, pp. 76–91, 2014.

[9] L. Langer, A. Schmidt, J. Buchmann, and M. Volkamer, "A taxonomy refining the security requirements for electronic voting: analyzing helios as a proof of concept," in *Availability, Reliability, and Security, 2010. ARES'10 International Conference on*. IEEE, 2010, pp. 475–480.

[10] V. Cortier, D. Galindo, S. Glondu, and M. Izabachene, "Election verifiability for helios under weaker trust assumptions," in *Computer Security-ESORICS 2014*. Springer, 2014, pp. 327–344.

[11] V. Cortier, D. Galindo, S. Glondu, and M. Izabachène, "Distributed elgamal á la pedersen: application to helios," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*. ACM, 2013, pp. 131–142.

[12] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology*. Springer, 1985, pp. 10–18.

[13] C.-P. Schnorr, "Efficient signature generation by smart cards," *Journal of cryptology*, vol. 4, no. 3, pp. 161–174, 1991.

[14] D. Chaum and T. P. Pedersen, "Wallet databases with observers," in *Advances in Cryptology–CRYPTO92*. Springer, 1992, pp. 89–105.

[15] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Advances in Cryptology–CRYPTO'97*. Springer, 1997, pp. 410–424.

[16] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology–CRYPTO94*. Springer, 1994, pp. 174–187.

[17] J. Camenisch, M. Stadler, J. Camenisch, and J. Camenisch, *Proof systems for general statements about discrete logarithms*. Citeseer, 1997.

[18] D. Bernhard, O. Pereira, and B. Warinschi, "How not to prove yourself: Pitfalls of the fiat-shamir heuristic and applications to helios," in *Advances in Cryptology–ASIACRYPT 2012*. Springer, 2012, pp. 626–643.

[19] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology–CRYPTO91*. Springer, 1992, pp. 129–140.

[20] S. Bayer and J. Groth, "Efficient zero-knowledge argument for correctness of a shuffle," in *Advances in Cryptology–EUROCRYPT 2012*. Springer, 2012, pp. 263–280.

[21] B. Terelius and D. Wikström, "Proofs of restricted shuffles," in *Progress in Cryptology–AFRICACRYPT 2010*. Springer, 2010, pp. 100–113.

[22] R. Kusters, T. Truderung, and A. Vogt, "Clash attacks on the verifiability of e-voting systems," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 395–409.

[23] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "Sok: A comprehensive analysis of game-based ballot privacy definitions," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 499–516.

[24] J. Benaloh, "Simple verifiable elections." *EVT*, vol. 6, pp. 5–5, 2006.

[25] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," DTIC Document, Tech. Rep., 2004.

[26] V. Cortier, G. Fuchsbauer, and D. Galindo, "Beleniosrf: A strongly receipt-free electronic voting scheme," Cryptology ePrint Archive, Report 2015/629, 2015, http://eprint.iacr.org/.

[27] O. Kulyk, V. Teague, and M. Volkamer, "Extending helios towards private eligibility verifiability," in *E-Voting and Identity*. Springer, 2015, pp. 57–73.

[28] M. V. David Bernhard, Oksana Kulyk, "Security proofs for participation privacy and stronger verifiability for helios," Cryptology ePrint Archive, Report 2016/431, 2016, http://eprint.iacr.org/.

[29] D. Demirel, J. Van De Graaf, and R. S. dos Santos Araújo, "Improving helios with everlasting privacy towards the public." in *EVT/WOTE*. USENIX, 2012.

[30] G. Tsoukalas, K. Papadimitriou, P. Louridas, and P. Tsanakas, "From helios to zeus," in *EVT/WOTE*. USENIX, 2013.

[31] P. Y. A. Ryan, P. B. Roenne, and V. Iovino, "Selene: Voting with transparent verifiability and coercion-mitigation," Cryptology ePrint Archive, Report 2015/1105, 2015, http://eprint.iacr.org/.

[32] S. Guasch and P. Morillo, "How to challenge and cast your e-vote," in *20th international conference on Financial Cryptography and Data Security*. Springer, 2016.

[33] A. Escala, S. Guasch, J. Herranz, and P. Morillo, "Universal cast-as-intended verifiability," in *20th international conference on Financial Cryptography and Data Security*. Springer, 2016.

[34] O. Kulyk, S. Neumann, M. Volkamer, C. Feier, and T. Koster, "Electronic voting with fully distributed trust and maximized flexibility regarding ballot design," in *Electronic Voting: Verifying the Vote (EVOTE), 2014 6th International Conference on*. IEEE, 2014, pp. 1–10.

[35] S. Neumann, M. M. Olembo, K. Renaud, and M. Volkamer, "Helios verification: To alleviate, or to nominate: Is that the question, or shall we have both?" in *Electronic Government and the Information Systems Perspective*. Springer, 2014, pp. 246–260.

[36] F. Karayumak, M. Kauer, M. M. Olembo, T. Volk, and M. Volkamer, "User study of the improved helios voting system interfaces," in *Socio-Technical Aspects in Security and Trust (STAST), 2011 1st Workshop on*. IEEE, 2011, pp. 37–44.

[37] A. Tchorbadjiiski, "Liquid democracy diploma thesis," *RWTH AACHEN University, Germany*, 2012.

[38] B. Zwattendorfer, C. Hillebold, and P. Teufl, "Secure and privacy-preserving proxy voting system," in *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference on*. IEEE, 2013, pp. 472–477.

[39] O. Spycher, R. Koenig, R. Haenni, and M. Schläpfer, "A new approach towards coercion-resistant remote e-voting in linear time," in *15th international conference on Financial Cryptography and Data Security*. Springer, 2011, pp. 182–189.

[40] R. Koenig, R. Haenni, and S. Fischli, "Preventing board flooding attacks in coercion-resistant electronic voting schemes," in *Future Challenges in Security and Privacy for Academia and Industry*. Springer, 2011, pp. 116–127.

[41] R. Haenni and R. E. Koenig, "A generic approach to prevent board flooding attacks in coercion-resistant electronic voting schemes," *Computers & Security*, vol. 33, pp. 59–69, 2013.